

## Evaluation of TCP Congestion Control Algorithms on the Windows Vista Platform\*

Yee-Ting Li

Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309

### Abstract

CTCP, an innovative TCP congestion control algorithm developed by Microsoft, is evaluated and compared to HSTCP and StandardTCP<sup>†</sup>. Tests were performed on the production Internet from Stanford Linear Accelerator Center (SLAC) to various geographically located hosts to give a broad overview of the performances.

We find that certain issues were apparent during testing (not directly related to the congestion control algorithms) which may skew results. With this in mind, we find that CTCP performed similarly to HSTCP across a multitude of different network environments.

However, to improve the fairness and to reduce the impact of CTCP upon existing StandardTCP traffic, two areas of further research were investigated. Algorithmic additions to CTCP for burst control to reduce the aggressiveness of its *cwnd* increments demonstrated beneficial improvements in both fairness and throughput over the original CTCP algorithm.

Similarly,  $\gamma$  auto-tuning algorithms were investigated to dynamically adapt CTCP flows to their network conditions for optimal performance. Whilst the effects of these auto-tuning algorithms when used in addition to burst control showed little to no benefit to fairness nor throughput for the limited number of network paths tested, one of the auto-tuning algorithms performed such that there was negligible impact upon StandardTCP. With these improvements, CTCP was found to perform better than HSTCP in terms of fairness and similarly in terms of throughput under the production environments tested.

---

\*Work supported by Department of Energy contract DE-AC02-76SF00515.

<sup>†</sup>The use of StandardTCP and NewReno are used interchangeably throughout this report, but generally, we define StandardTCP as the stack implementation of the NewReno congestion control algorithm.

# 1 Introduction

Windows Vista provides facilities to switch TCP congestion control algorithms. This report investigates the implementation and performance of these algorithms on real WAN networks from SLAC to various locations worldwide. In particular, the report focuses on CTCP [2, 1], a Microsoft developed congestion control algorithm that marries both the Loss Based nature of standard TCP and the Delay Based features of TCP Vegas.

Due to time constraints of this report, a full investigation on the various CTCP performance details was not possible. As such, this report serves as an initial field study of the applicability of deploying CTCP as the default TCP congestion control algorithm in Windows Vista.

This applicability is considered in the following forms; raw throughput achieved by CTCP flows, the fairness between both CTCP flows competing with one another (intra-protocol fairness) and that when competing against the default TCP congestion algorithms (friendliness). These *metrics* are compared to that with using the default NewReno stack and also that of an implementation of HSTCP [6] so that a comparative judgement upon relative performance can be made.

## 2 Test Outline

As per [4], we focus upon the following metrics for the direct comparison between different TCP algorithms.

- Intra-Protocol Fairness

In this scenario, we vary the number of TCP flows from 2 to  $n$ . The fair share of network bandwidth between the flows is vital to ensure fair sharing of network resources and good scaling properties of the TCP algorithm.

- Convergence Time

The convergence of fairness towards some arbitrary constant ratio is important to understand the dynamic transient fairness between TCP flows. This is especially useful in understanding the interplay between a newly started flow and that of flows that are already running.

- Inter-Protocol Fairness & Friendliness

The fairness between the various TCP algorithms. In particular, the scaling properties of fairness between the New-TCP algorithms and that of Standard TCP is important to ensure smooth deployment of new congestion control algorithms.

- Reverse Traffic Load

A well recognised problem with Loss Based algorithms is caused by the RTT delay variation as a result of loading on the reverse data path. The queuing associated with the reverse path traffic may cause the TCP algorithm to act inappropriately and thus reduce performance.

	Specification
Vendor & Part	Dell 2650
CPU	3Ghz
Memory	2GB
NIC	Neterion XFrame I
NIC Driver	2.6.12.0

Table 1: PC configuration at SLAC.

Destination	Location	RTT	No. Hosts
Caltech	US West Coast	8.5msec	1
Florida	US East Coast	71.0msec	2
Dublin	Ireland, International	151msec	2

Table 2: Destination Sites and Configuration of PC's at those sites.

## 2.1 Parameters Not Investigated

The fairness between flows of different end-to-end latencies (RTT Fairness) was identified in [4] as being an important factor in the scalability of TCP algorithms as low latency flows may starve long latency (and hence less aggressive) flows. Important deductions from the literature demonstrates that severe RTT unfairness between the various TCP algorithms is apparent. However, due to the requirement of a common bottleneck and the difficulty in engineering such a topology on the Internet, these sets of experiments were not conducted.

## 3 Test Setup

### 3.1 Hosts and Equipment

Microsoft Windows Vista build 5270 (December 2005) was installed on two machines (See Table 1) at SLAC and were connected onto the commodity production network: ESnet at 1Gbit/sec and Internet2 (via Stanford University) also at 1Gbit/sec via a Cisco 6506.

Initial tests using the internal Broadcom network interface cards on these machines showed serious reordering (most likely a driver issue) and therefore the default NIC's were changed to Neterion XFrame I cards capable of 10Gb/sec. The newest drivers for this card were back ported (courtesy of Neterion) onto build 5270.

The PC's were connected to a dedicated Cisco 6909 using Cisco SR Xenpaks, and a 1Gbit/sec connection was made between the 6909 and the 6906 to provide production network access.

Three remote hosts were identified to be capable of high performance from which tests were conducted to and are shown in Table 2 and Table 3. Note that only one machine was available at Caltech and the hosts at Florida are only Pentium III class servers.

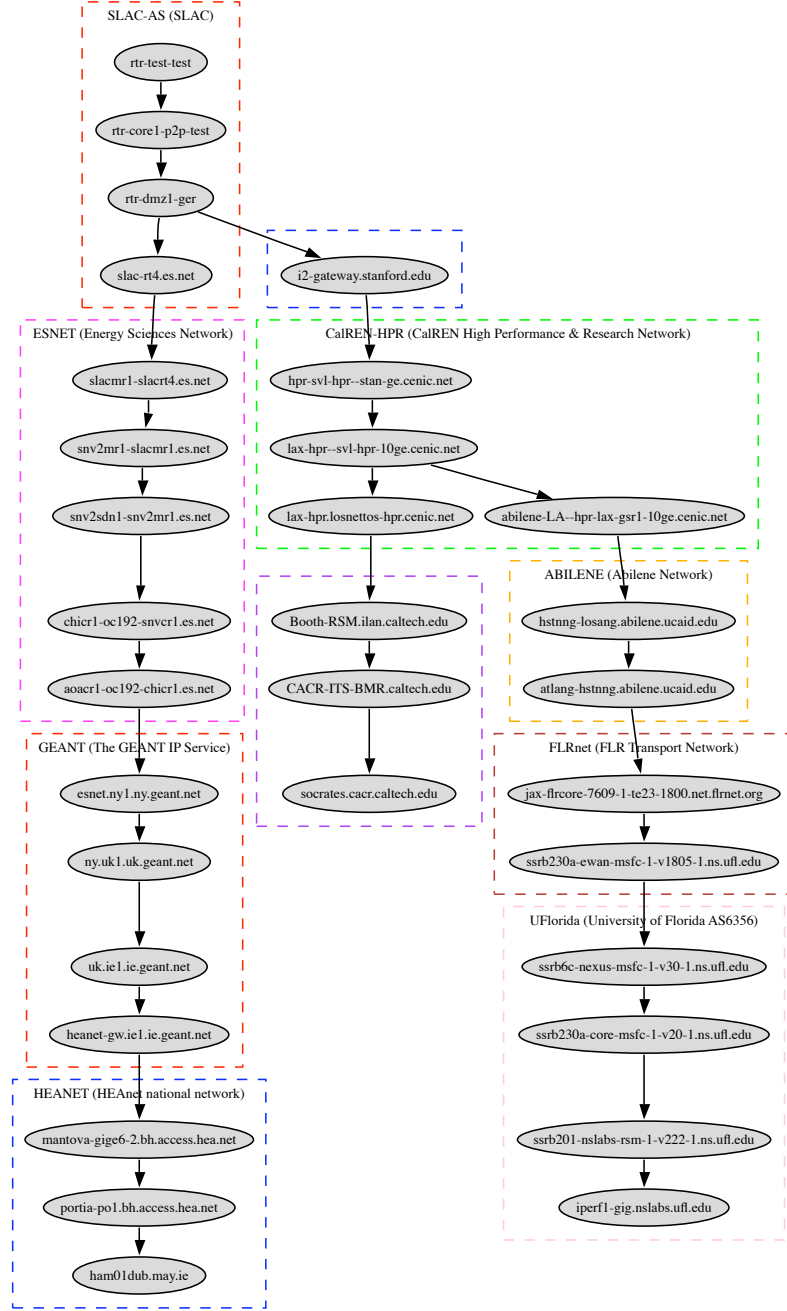


Figure 1: Traceroute graph from SLAC to the Various Destination Hosts.

Destination	CPU	Memory	Network Interface Card
Caltech	4×2.8Ghz Xeon	1GB	SysKonnnect SK-98xx Gigabit Ethernet Server Adapter
Florida	1Ghz Pentium III	0.5GB	Syskonnnect (Schneider & Koch) Gigabit Ethernet
Dublin	2.4Ghz Xeon	4GB	Intel Corp. 82545EM Gigabit Ethernet Controller

Table 3: Network Interface Cards at Destination Sites

The above hosts remain on relatively stable network connections with virtually zero route flapping. The route through the Internet to each host is shown in Figure 1. The background traffic experienced for all hosts were primarily academic (Internet2) and or national lab (ESnet) network traffic.

### 3.2 Testing Methodology

In order to maximise the raw number of results and determine with confidence the relative performances of each TCP congestion control algorithm to each destination, tests were ran continuously, in a random fashion, 24 hours a day, 7 days a week. However, the results shown in Tables 4 to 7 were collated over the final week of February 2006.

The tests were automated using custom scripts (tools-server) from SLAC which enabled ‘remote control’ of network nodes to start and gather test results. The automation required some adaptation onto Windows hosts, and thus required ActivePerl and IIS to be installed to facilitate the running of tests and the retrieval of test results respectively. This automation of tests enabled off-line (i.e. not live) analysis of test results.

All tests were ran with iperf [3], with custom modifications by Microsoft Research China to allow the dynamic switching of the TCP congestion control algorithm using the ‘-q’ argument.

The duration of each test depends upon the type of test in question, but typically, sufficient time was given to provide the interaction between TCP flows to stabilise. Tests were chosen to run between 15 minutes and 45 minutes depending upon the complexity of the test. Individual flows in each test were ran in a staggered fashion such that the convergence times between flows can be calculated and the dynamic interaction between TCP flows can be seen. For tests which were staggered, the summary results only shows the performance when all flows were being ran; i.e. for 8 flow tests, the calculated statistics only represent the period whereby all 8 flows were competing.

### 3.3 Test Cases

A series of simple test cases were investigated in order to determine the properties of the various TCP algorithms under investigation.

- Single Flow

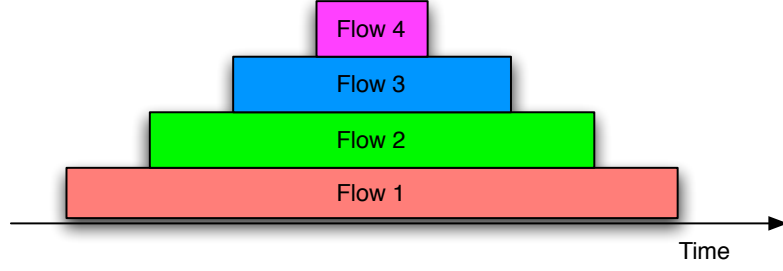


Figure 2: The time based initiation and termination of flows.

A single TCP flow will be initiated with CTCP, StandardTCP or HSTCP for a duration of 15 minutes to one of the 3 destinations.

This test will investigate the raw throughput performance and correct implementation of the TCP algorithms. It may also identify potential problems or beneficial features in the design of the Windows TCP stack.

- Single Flow with Many TCP Flows Reverse Traffic

A single TCP flow will be initiated, for a duration of 15 minutes. During this time,  $n$  number of standard TCP flows are run in the opposing direction.

The effect of this tests is to determine the interaction between the TCP algorithm and the effects of increased latency on the reverse path, and also potentially the effects of stretched acks.

- $n$ -Flows

A single TCP flow will be initiated, and at a subsequent time later (currently 5 minutes), a second TCP flow will be initiated from the other host. If  $n$  is greater than 2, then at a subsequent time later (another 5 minutes), another flow will be initiated.

The tests are engineered such that  $n$  flows will compete for bandwidth for at least 15 minutes before the flows are stopped in reverse order in 5 minute steps until all flows are terminated. This is illustrated in Figure 2.

Depending upon the type of congestion control algorithm used, this test will be used to determine the inter or intra-protocol fairness between the stacks. The metric of fairness is defined in Section 4.

When at least one of the TCP flows is StandardTCP, the test is considered as a Friendliness test.

To decrease the possibility of host interactions, it would be preferable to have a single flow per machine; however, due to limited hardware constraints, the flows will be balanced between the two available hosts.

- Eight Flows

Pairs of TCP connections will be initiated in 5 minute intervals in the same fashion as with the  $n$ -Flows case (instead of a single flow). That is at time 0, two flows will be initiated on host 1, and then five minutes later two more flows will be initiated on host 2. 10 minutes after the start of the test, another 2 flows are initiated from host 1 and finally 5 minutes after this, the final pair of flows are initiated on host 2.

## 4 Metrics

Two primary metrics are considered; the *fairness* and the *throughput* of the TCP algorithms.

We consider the throughput<sup>‡</sup>  $G$  to be the summed mean throughput of each flow,  $i$ , i.e.  $\bar{x}_i$  (See Equation 1).

$$G := \sum_{i=1}^n \bar{x}_i \quad (1)$$

We consider the following two metrics to capture the complex metric of fairness between many flows.

- $\sigma$  fairness:

We define  $\sigma_f$  to be the standard deviation from the mean throughput of all flows  $\bar{x}$  normalised by the mean throughput  $\bar{x}$ , where  $\bar{x}_i$  is the mean throughput of the  $i^{th}$  flow,:

$$\sigma_f := \frac{1}{\bar{x}} \sqrt{\frac{\sum_{i=1}^n (\bar{x}_i - \bar{x})^2}{n}} \quad (2)$$

The properties of this metric define the magnitude of the differences between the throughputs of each flow<sup>§</sup>. Therefore, it is preferable to have a low value of  $\sigma_f$ , whereby  $\sigma_f \rightarrow 0$  means perfect fairness between all flows.

- $\xi$  fairness:

In order to gauge the dynamics between the flows as they interact through time, we define  $\xi_f$  as the standard deviation of the standard deviations  $\sigma_t$  of throughputs  $x_i$  at time interval  $t$  for all time  $T$  - normalised by the mean throughput  $\bar{x}$ :

$$\xi_f := \frac{1}{\bar{x}} \sqrt{\frac{\sum_{t=1}^T (\bar{\sigma}_t - \bar{\sigma})^2}{T}} \quad (3)$$

Note that in many regards,  $\xi_f$  can be considered as compound metric that captures both the nature of fairness and the transient fairness (in terms of differences in throughput magnitude) between flows.

---

<sup>‡</sup>Specifically, we do not distinguish between *goodput* and *throughput*, i.e. we assume reduction in bytes transferred per unit time due to retransmissions to be negligible.

<sup>§</sup>Note that this differs from the commonly defined *stability* of a single flow as we do not consider a time series of throughputs, but the mean throughput of *all* flows under investigation

Test	TCP Algorithm	Caltech	Florida	Ireland
1 Flow	StandardTCP	3	3	4
1 Flow	CTCP	6	4	3
1 Flow	HSTCP	4	4	3
2 Flows	StandardTCP	4	6	8
2 Flows	CTCP	5	17	11
2 Flows	HSTCP	3	8	9
4 Flows	StandardTCP	3	2	3
4 Flows	CTCP	2	5	5
4 Flows	HSTCP	2	4	2
8 Flows	StandardTCP	3	2	2
8 Flows	CTCP	2	6	11
8 Flows	HSTCP	2	3	6
16 Reverse TCP	StandardTCP	3	11	2
16 Reverse TCP	CTCP	2	10	13
16 Reverse TCP	HSTCP	3	7	15

Table 4: Number of individual Tests performed.

The specific differences between  $\xi_f$  and  $\sigma_f$  incorporate the time sensitive nature of convergence and temporal fairness between flows. A specific example of this is that if all flows show zero variation in throughput along time,  $\xi_f \rightarrow \sigma_f$ .

## 5 Results

Table 4 shows the number of individual completed tests for each type of experiment. Due to time constraints, and the nature of Internet traffic, a full statistical study of the performance of the algorithms was not possible. However, in an attempt to capture the confidence levels of the tests, the standard error of the measurements are presented - however, due to the rather low sample sizes the accuracy of the test information should be ‘taken with a pinch of salt’.

Table 5 shows the aggregate throughput  $G$  of all of the tests. Note that due to the diurnal nature of Internet traffic, and the low number of samples taken for each test, that the throughputs achieved are not conclusive. In particular, certain tests shows that StandardTCP actually achieves a higher mean throughput than the other algorithm; however, it also has a very large standard error - a consequence of the time at which the tests were ran. Therefore, by running more tests over a longer time frame, it should be possible to accurately capture the performance of the algorithms under a wide variety of network conditions.

However, generally the results show that CTCP is statistically equivalent to HSTCP in terms of throughput. An exception this this was under the Florida path where with 1 or two flows, CTCP is able to achieve twice as much throughput as that of HSTCP. This difference between the different TCP algorithms becomes less noticeable with more flows as CTCP and HSTCP begin to fall back into congestion control mechanisms similar to that



Test	TCP Algorithm	Caltech	Florida	Ireland
1 Flow	StandardTCP	521±15	114±5	62±19
1 Flow	CTCP	619±21	252±13	146±37
1 Flow	HSTCP	605±21	125±7	135±20
2 Flows	StandardTCP	574±14	203±19	167±35
2 Flows	CTCP	640±11	347±3	190±22
2 Flows	HSTCP	504±136	161±7	216±42
4 Flows	StandardTCP	645±26	280±3	223±42
4 Flows	CTCP	653±33	379±1	258±42
4 Flows	HSTCP	668±32	263±51	253±160
8 Flows	StandardTCP	732±74	331±1	431±75
8 Flows	CTCP	672±23	389±2	435±53
8 Flows	HSTCP	636±37	392±3	497±29
16 Reverse TCP	StandardTCP	96±9	33±1	128±14
16 Reverse TCP	CTCP	108±7	45±2	121±11
16 Reverse TCP	HSTCP	87±12	114±4	133±10

Table 5: Aggregate throughput (mbit/sec)  $G$  of Tests.

Test	TCP Algorithm	Caltech	Florida	Ireland
1 Flow	StandardTCP	-	-	-
1 Flow	CTCP	-	-	-
1 Flow	HSTCP	-	-	-
2 Flows	StandardTCP	0.023±0.012	0.292±0.148	0.118±0.030
2 Flows	CTCP	0.006±0.004	0.062±0.020	0.305±0.063
2 Flows	HSTCP	0.076±0.035	0.496±0.149	0.292±0.075
4 Flows	StandardTCP	0.109±0.082	0.024±0.001	0.107±0.031
4 Flows	CTCP	0.017±0.008	0.032±0.010	0.362±0.061
4 Flows	HSTCP	0.091±0.022	0.622±0.311	0.410±0.015
8 Flows	StandardTCP	0.114±0.048	0.048±0.000	0.169±0.048
8 Flows	CTCP	0.013±0.006	0.028±0.005	0.345±0.016
8 Flows	HSTCP	0.090±0.013	0.245±0.180	0.386±0.029
16 Reverse TCP	StandardTCP	-	-	-
16 Reverse TCP	CTCP	-	-	-
16 Reverse TCP	HSTCP	-	-	-

Table 6:  $\sigma_f$  of Tests.

Test	TCP Algorithm	Caltech	Florida	Ireland
1 Flow	StandardTCP	-	-	-
1 Flow	CTCP	-	-	-
1 Flow	HSTCP	-	-	-
2 Flows	StandardTCP	0.214±0.007	0.554±0.202	0.289±0.040
2 Flows	CTCP	0.145±0.005	0.210±0.020	0.441±0.035
2 Flows	HSTCP	0.540±0.222	1.000±0.065	0.458±0.045
4 Flows	StandardTCP	0.294±0.067	0.236±0.003	0.256±0.031
4 Flows	CTCP	0.174±0.010	0.208±0.009	0.549±0.035
4 Flows	HSTCP	0.432±0.005	0.961±0.350	0.519±0.035
8 Flows	StandardTCP	0.326±0.037	0.285±0.022	0.317±0.032
8 Flows	CTCP	0.236±0.018	0.223±0.002	0.570±0.029
8 Flows	HSTCP	0.677±0.005	0.434±0.148	0.597±0.058
16 Reverse TCP	StandardTCP	-	-	-
16 Reverse TCP	CTCP	-	-	-
16 Reverse TCP	HSTCP	-	-	-

Table 7:  $\xi_f$  of Tests.

of StandardTCP. This occurs under environments of high packet loss and when network congestion is detected for HSTCP and CTCP respectively.

Tests involving reverse TCP background traffic are discussed in-depth in Section 6.2.

Table 6 shows the  $\sigma_f$  of the flows. They generally show that HSTCP has a relatively larger value of  $\sigma_f$  which is discussed in Section 6.4. The results for CTCP show that under the Caltech and Florida paths, CTCP shows statistically lower values of  $\sigma_f$  than both StandardTCP and HSTCP. However, under the long distance Ireland path, CTCP's  $\sigma_f$  is comparable to that of HSTCP.

$\xi_f$  is shown in Table 7. Results indicate that the CTCP generally has a lower  $\xi_f$  value than that of both HSTCP and StandardTCP, except under the the Ireland path where both HSTCP and CTCP show equivalently larger values of  $\xi_f$  when compared to StandardTCP.

## 6 Case Studies

The following case studies demonstrate the various application of the metrics above, and highlights problems that were experienced with the tests. Unless otherwise indicated, these case studies show only anomalous behaviour and should not be taken as indicative of the test results shown.

### 6.1 High Losses Resulting in Consecutive Backoffs

Common to all test results, especially those with flows at high throughput, is the occurrence of what appears to be multiple reductions in throughput - which causes a severe reduction in the measured average throughput of the flow(s). This is shown in Figure 3. As the occurrence

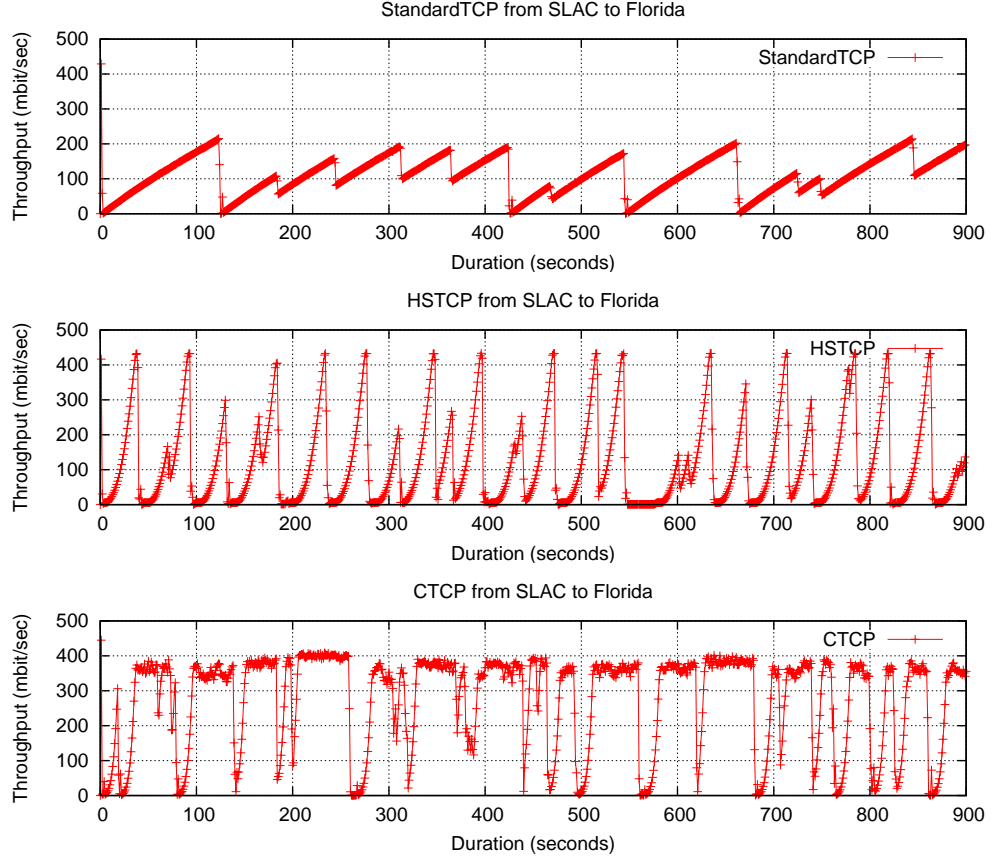


Figure 3: Dramatic Drops experienced by all TCP algorithms.

of these large drops in throughput is apparent on all TCP algorithms, it is assumed that the problems are associated with the network stack rather than with the congestion control algorithms.

Figure 4 shows the *cwnd*, *ssthresh* and *dwnd* of the CTCP graph in Figure 3. Note that after a loss, the value of *ssthresh* should be set to  $b \times cwnd$ <sup>¶</sup> (where  $b = 0.5$  under StandardTCP and CTCP), and then subsequently *cwnd* should be set to this value of *ssthresh*. The purpose of this is so that TCP slow start can quickly regain the lost throughput should a timeout occur. However, Figure 4 shows that in fact the *ssthresh* value is sometimes set to very small values (sometimes from thousands of packets down to only tens of packets).

Figure 5 shows the *cwnd* and *ssthresh* dynamic of the altAIMD Linux kernel [4] and with Windows Vista with their implementations of HSTCP. Due to the temporal differences between the tests, a direct comparison between the two implementations can not be made directly from these graphs (e.g. the relatively smaller values of the Linux implementation could be due to excess cross traffic). However, it can be seen that under the Windows implementation at approximately 350 and 650 seconds that the *ssthresh* value after a congestion

<sup>¶</sup>In fact, it *ssthresh* should be set to  $b \times f$  where  $f$  is equal to the number of packets ‘in-flight’, which to first approximation is typically approximately the same as *cwnd*.

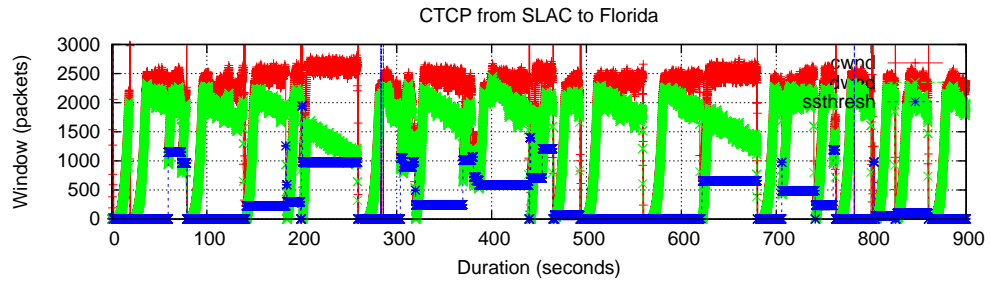


Figure 4: Drops experienced by CTCP algorithm.

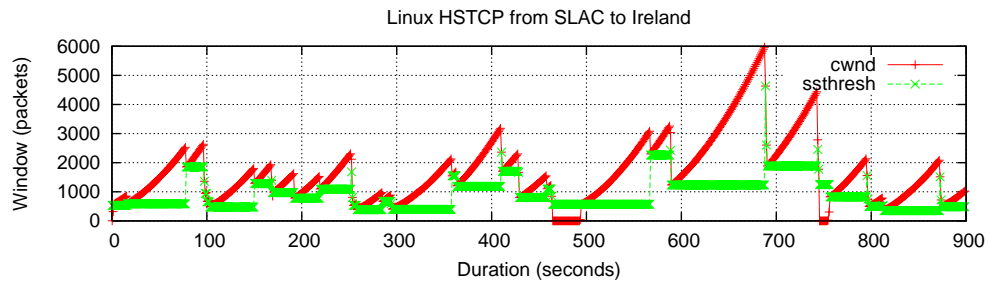
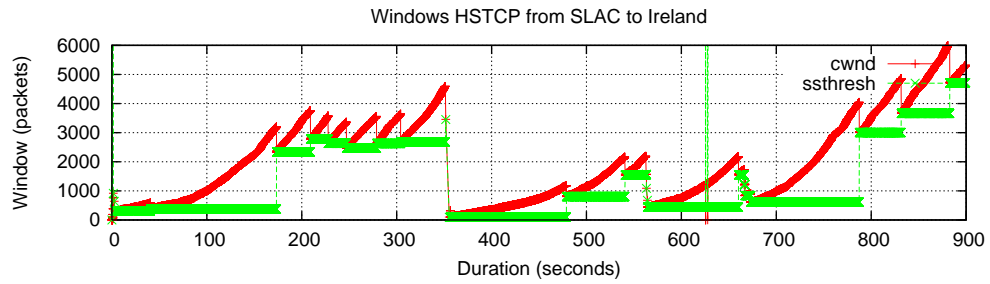


Figure 5: Drops experienced by Linux and Windows HSTCP algorithm.

```

12:40:15.610 [tcp]dup halve. DWnd = 1149703 CWnd = 1311918 sndUna = 3870992794
12:40:16.058 [tcp]CTcpTimeoutCallback: CWnd = 7097898 SsThresh = 1307538
12:40:16.058 [tcp]dup halve. DWnd = 0 CWnd = 7300 sndUna = 3871619134
12:40:16.132 [tcp]CTcpTryToOpenDWnd: CWnd = 1084783 SsThresh = 2920, BaseRTT 0
12:40:16.210 [tcp]CWnd Compensated! CWnd = 6603580 DWnd = 1460 SndUna = 3872141814
12:40:16.210 [tcp]dup halve. DWnd = 730 CWnd = 3306170 sndUna = 3872141814
12:40:16.714 [tcp]CTcpTimeoutCallback: CWnd = 4447890 SsThresh = 3301790
12:40:16.857 [tcp]dup halve. DWnd = 0 CWnd = 10950 sndUna = 3872375414
12:40:17.433 [tcp]dup halve. DWnd = 0 CWnd = 7665 sndUna = 3878745394
12:40:17.505 [tcp]Tcp Rtt Samples: CWnd = 3285, SsThresh = 3285, Rtt = 6
12:40:17.643 [tcp]Tcp Rtt Samples: CWnd = 8818, SsThresh = 3285, Rtt = 8
12:40:17.714 [tcp]Tcp Rtt Samples: CWnd = 10189, SsThresh = 3285, Rtt = 8

```

Figure 6: TCP Trace of grepped events leading to Consecutive Backoffs

event falls to very small values which in turn prevents the use of slow-start after time-outs and forces the congestion control algorithm to start from very low *cwnd* values. Whilst multiple drops are also apparent in the Linux implementation, it rarely forces *ssthresh* to such small values as suddenly as it does for the Windows implementation.

Given that the TCP logging in Windows is event based, these large drops in *ssthresh* in sequential events is unexpected: an inspection of the TCP event trace shows that there is a complex interaction of *dup halves* and *CTcpTimeoutCallback*'s which causes the window values to fall. Further investigation of this is required to fully understand the interaction between what is actually happening with regards to packet events to what actually occurs in the TCP stack. A snapshot of the trace is shown in Figure 6.1.

## 6.2 The Effect of Reverse Traffic

There is a requirement for sufficient delay information for CTCP to operate efficiently. Traffic on the reverse path of the TCP flow will increase the measured latency resulting in reduced performance for the CTCP flow.

Table 5 indicates that there is little effect upon the throughput of reverse traffic on the Ireland link. However, there is a very pronounced effect on the Caltech and Florida paths - for all TCP algorithms, not just CTCP. Whilst the typical single flow tests demonstrate throughputs in excess of 500Mbit/sec for all TCP algorithms; in the presence of 16 TCP flows in the reverse direction, attainable throughput diminishes to approximately 100Mbit/sec. Similarly, the Florida link sees a reduction from 100-200Mbit/sec down to about 40Mbit/sec.

There is an exception with HSTCP on the Florida link whereby similar throughputs are achieved with and without reverse traffic. Further research is needed to be conducted to deduce this discrepancy.

Whilst some of the problems of reduced throughput (compared to having no reverse traffic) may be explained by the fact that Caltech only has a single host, the other tests were performed with the reverse traffic on a separate pair of hosts from that of the test TCP flow such that host issues (e.g. high flux of interrupts, memory and CPU saturation, etc.) should not affect the results.

An important point to note here is that in the presence of reverse traffic, the throughput of CTCP is no worse than that of StandardTCP. However, some further investigation is required to determine the cause of the reduced forward path throughput for the Florida and Caltech path.

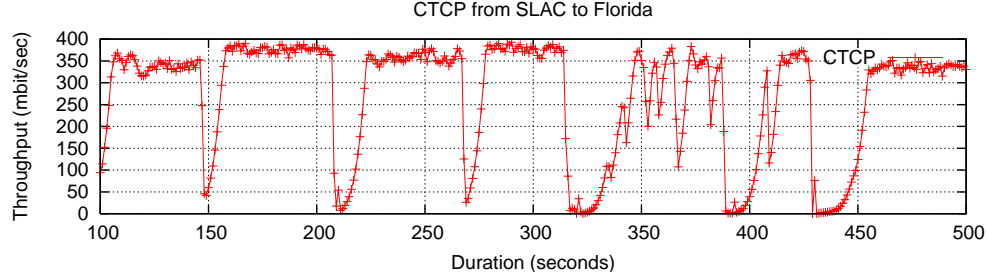


Figure 7: Effect of CTCP Delay Based Congest Control with sufficient Queuing.

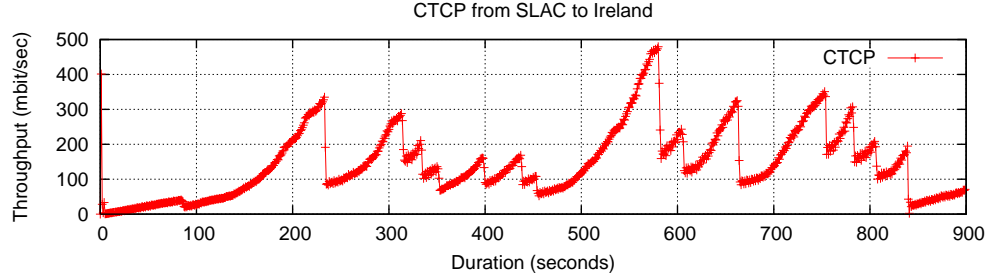


Figure 8: Effect of CTCP Delay Based Congest Control with *insufficient* Queuing.

### 6.3 The Effect of Small Queue's in CTCP

The CTCP algorithm utilises delay information in order to maximise throughput. A characteristic profile of this action is shown in Figure 7 which shows a CTCP transfer from SLAC to Florida. using trace data of the TCP flows, it was determined that the minimal and maximum latencies experienced results in a derived/calculated bottleneck queuesize of approximately 375 packets - of which CTCP achieves approximately 50 packets `diffWnd` in the network. As the default  $\gamma$  setting of 30 packets is smaller than `diffWnd`, the delayed based component of CTCP is able to work effectively.

However, when there is insufficient queuing available on the network link, the plateau whereby CTCP is able to maintain high throughput is lost, as shown in Figure 8. It can be clearly seen that CTCP is only able to achieve the binomial growth of throughput, and does not manage to determine sufficient delay information from the network.

Using the same technique as for the Florida path, the calculated bottleneck queuesize of the Ireland path is only 250 packets. Given the current defined  $\gamma$  setting of 30 packets, the queue should be able to hold at least 8 CTCP flows. However, given that the network is shared amongst many other Internet users, it was found through the TCP traces that the `diffWnd` value (i.e. the actual number of packets in the network 'pipe' for one flow) was only about 3 packets. Therefore, as `diffWnd` <  $\gamma$ , the CTCP flow is unable to utilise the network effectively.

Section 7 and 8 investigates new CTCP algorithmic additions to mitigate the effects of both having a static value of  $\gamma$  and to reduce the effect of large bursts as `diffWnd` approaches this value.

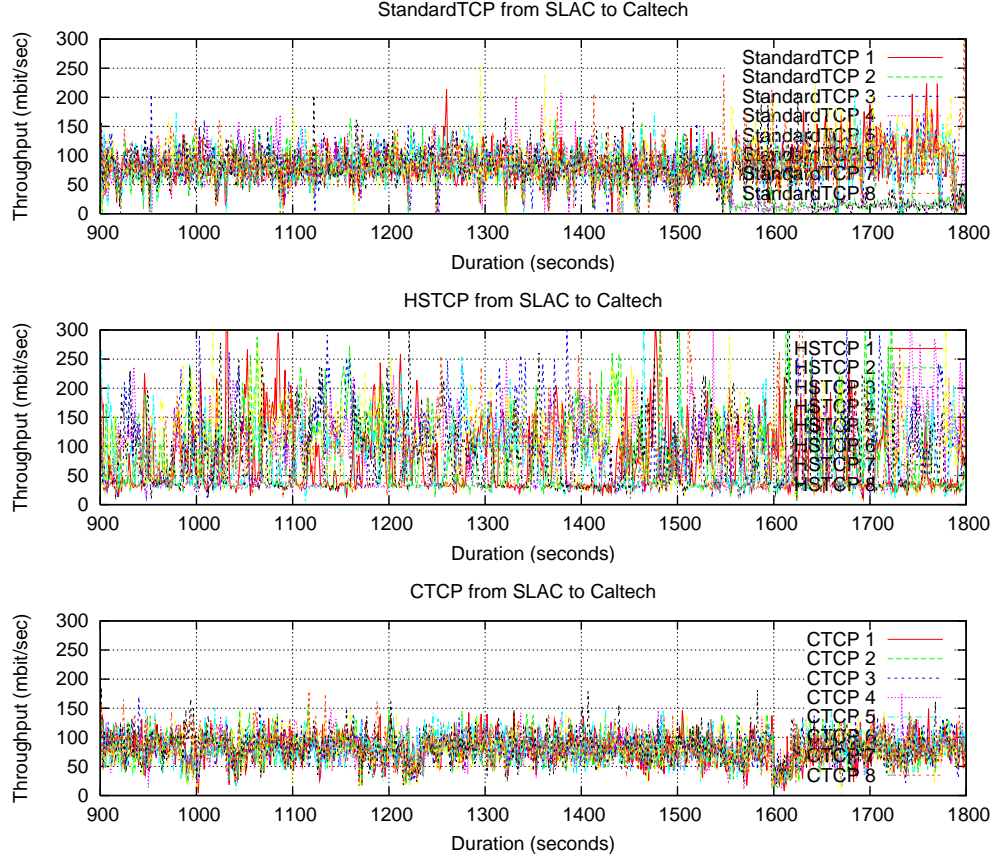


Figure 9: Throughput of individual flows from SLAC to Caltech.

## 6.4 High $\sigma_f$ of HSTCP

The main cause of the high  $\sigma_f$  values of HSTCP was found to be caused primarily by the large number of consecutive drops in many tests (See Section 6.1) which causes large fluctuations between HSTCP flows. This was the most apparent with a smaller number of flows.

Whilst the effect of the multiple consecutive drops is apparent on all TCP algorithms, the effect appears to be the most pronounced with HSTCP. This may be related to an aggressive increase in  $\alpha$  which causes large bursts of packets which may then be subsequently dropped by the network [5].

## 6.5 Low $\xi_f$ of CTCP

Figure 9 demonstrates the applicability of the  $\xi_f$  parameter. It shows the throughput of each individual flow in an 8-Flow test from SLAC to Caltech and demonstrates the wildly fluctuating throughput of HSTCP - as mentioned in Section 6.1, this was due to multiple consecutive drops. Results on Table 7 show that StandardTCP and CTCP have similar values of  $\xi_f$  - with CTCP having a slightly smaller value (and hence more ‘fair’).

Figure 10 shows the variability of the throughputs on a less well provisioned link and

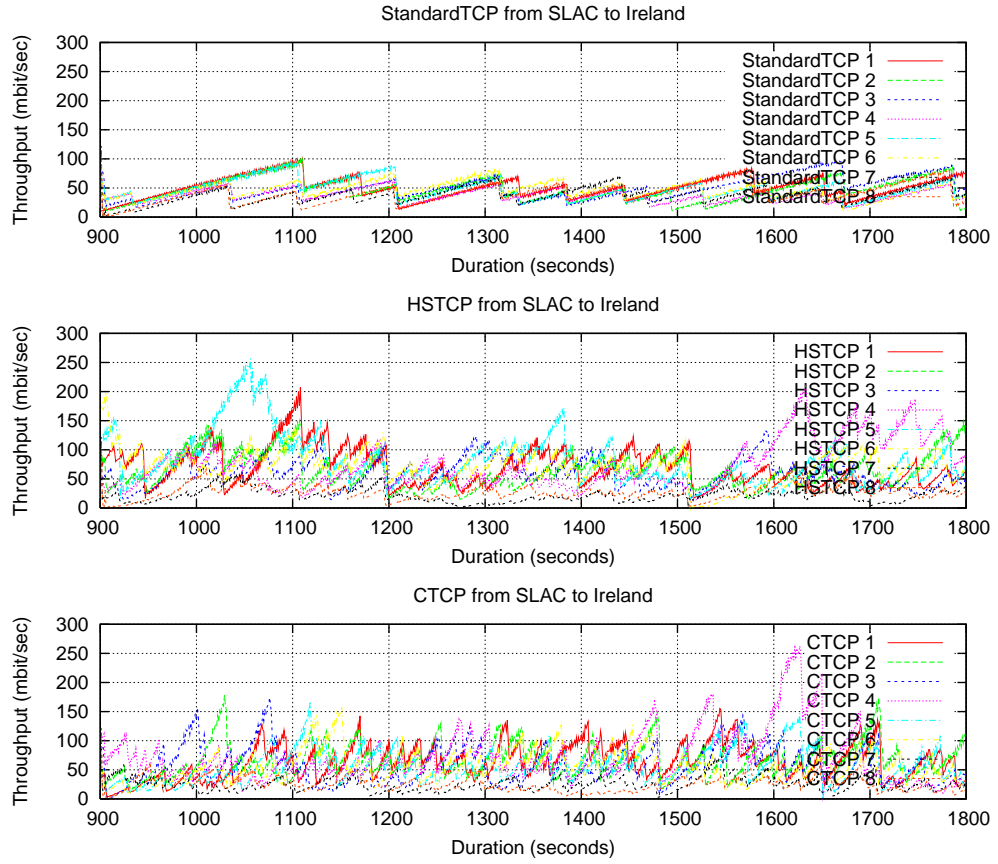


Figure 10: Throughput of individual flows from SLAC to Ireland.



Flows	CTCP Algorithm	Samples	Throughput	$\sigma_f$	$\xi_f$
1	Original	29	116±7	–	–
1	<i>with muted dwnd increments</i>	35	222±8	–	–
1	<i>with partial dwnd increments</i>	36	177±11	–	–
2	Original	19	143±11	0.22±0.08	0.52±0.06
2	<i>with muted dwnd increments</i>	15	197±24	0.13±0.02	0.34±0.02
2	<i>with partial dwnd increments</i>	18	217±26	0.10±0.02	0.35±0.01
8	Original	10	384±15	0.35±0.05	0.87±0.04
8	<i>with muted dwnd increments</i>	9	474±35	0.19±0.02	0.52±0.03
8	<i>with partial dwnd increments</i>	8	470±23	0.27±0.03	0.61±0.02

Table 8: Performance of CTCP with burst control from SLAC to Ireland.

demonstrates a similar profile to that of HSTCP. It may be interesting to investigate further into the performance of CTCP compared to that of HSTCP over a range of different environments as the delay based nature of CTCP may or may not result in fairer sharing of the bottleneck link for competing CTCP flows.

## 7 CTCP Burst Control

A couple of proposals within Microsoft aims to mitigate the effects of having a large static value of  $\gamma$  which has been shown to be detrimental to CTCP performance over small buffered links. They aim to reduce the large bursts of packets onto the network by utilising a different *cwnd* growth law between two parameters  $\gamma_{low}$  and  $\gamma$ .

Two of these burst control algorithms were implemented and tested over the SLAC-Dublin Internet path. This particular path was chosen as it demonstrated small *diffWnd* utilisation in previous tests where these modifications should be the most useful. The idea of the algorithms is to maintain a slower growth of *cwnd* when the measured *diffWnd* value is within the range  $\gamma_{low} \rightarrow \gamma$ . CTCP *with muted dwnd increments* attempts to do this by muting the *dwnd* increases, whilst CTCP *with partial dwnd increments* limits the growth rate of *dwnd* to some value. CTCP is the original CTCP algorithm as investigated in the previous tests.

Table 8 shows the difference in performance between the CTCP flows with and without the burst control with various numbers of concurrent CTCP flows from SLAC to Ireland. These tests were performed during the 2nd and 3rd week of April 2006.

### 7.1 Single Flow

It can be clearly seen that both burst control algorithms facilitate higher throughput than the original algorithm. This is due primarily to the evident throughput reduction of the original CTCP version towards zero with nearly every congestion event (see Figure 11).

Both CTCP with burst control additions show increased throughput performance over the original CTCP algorithm. CTCP *with muted dwnd increments* is clearly able to achieve

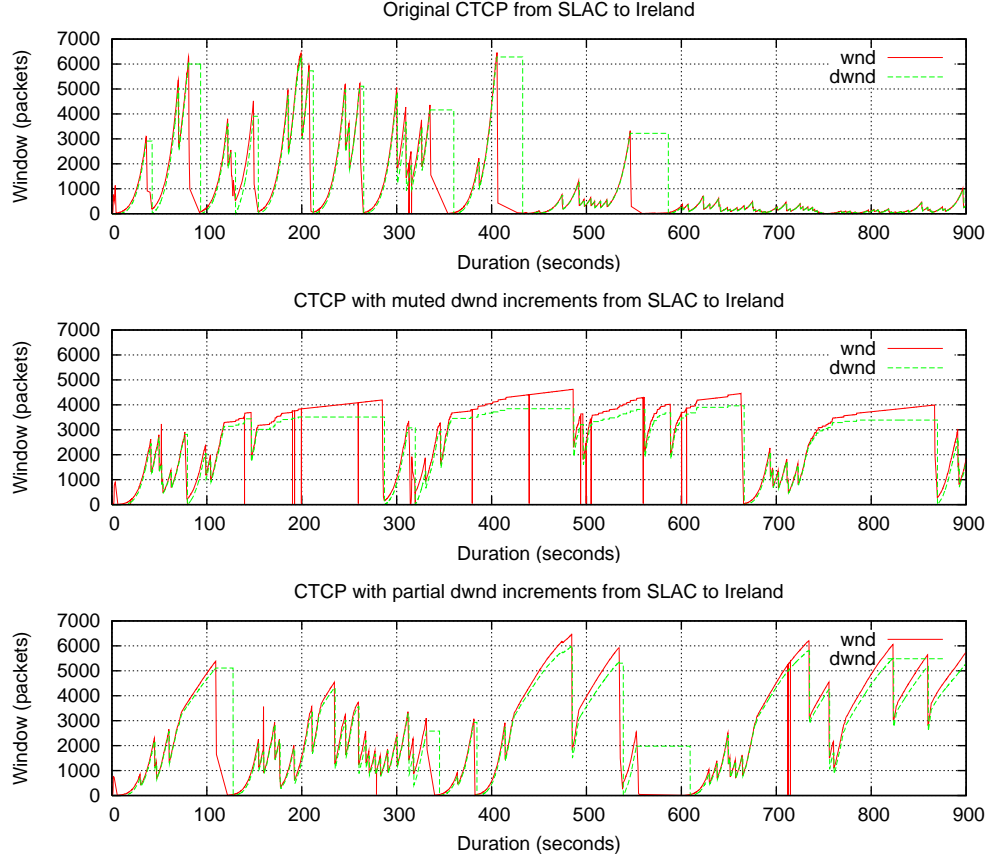


Figure 11: Window dynamic of CTCP with burst control from SLAC to Ireland.

the highest throughput, with approximately 20% higher throughput than that of CTCP *with partial dwnd increments*.

Figure 11 shows the different dynamics of the three CTCP algorithms in terms of their *wnd* and *dwnd* values. It can be seen that the reason why CTCP *with muted dwnd increments* is able to achieve such high throughputs is due to the *dwnd* values plateauing off - leaving only the NewReno algorithm to cause the expected induced loss. As the duration of sustained high throughput is longer, this increases the value of measured average throughput.

## 7.2 Two Flows

Again, both CTCP algorithms with burst control achieve greater throughput when compared to the original CTCP algorithm. It can be seen that the aggregate throughput of two flows is highest with the CTCP *with partial dwnd increments* variant - although the errors associated with the measurements indicate no significant difference between the two versions. Similarly, both new CTCP with burst control experience notably better fairness than the original algorithm.

Figure 12 shows the throughput dynamic between the two competing CTCP flows. It can be observed that even though CTCP *with muted dwnd increments* is able to sustain

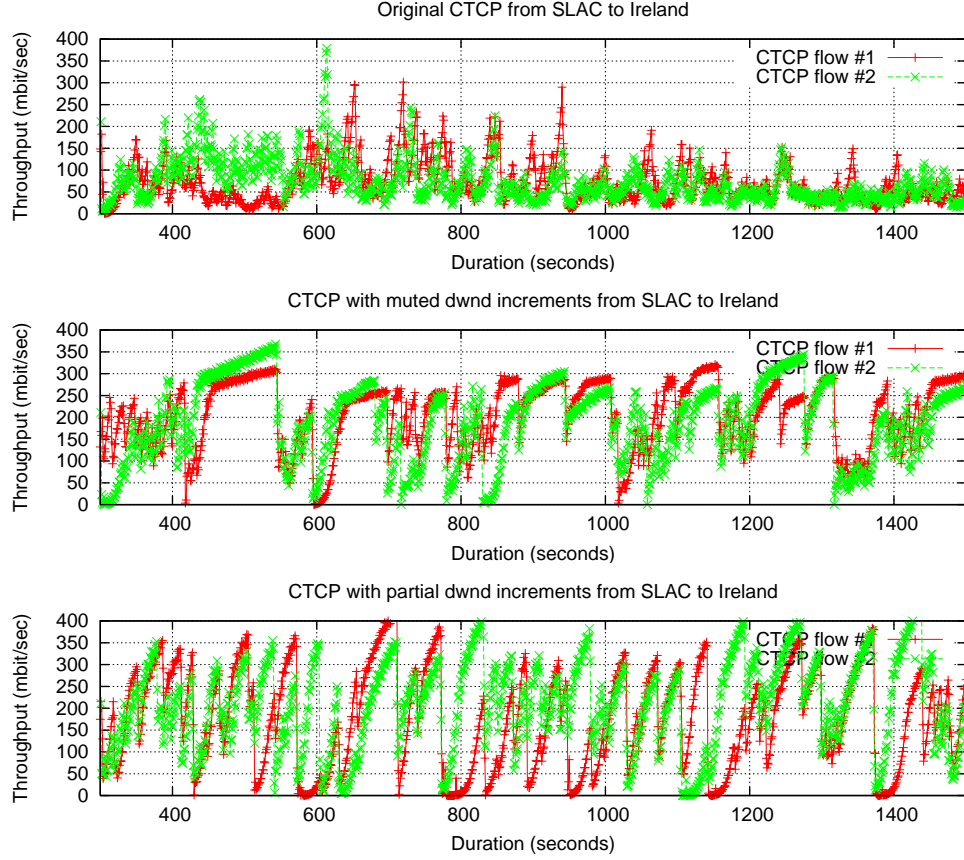


Figure 12: Interaction between CTCP with burst control from SLAC to Ireland.

longer periods of high throughput, the peak throughput is generally less than that of CTCP *with partial dwnd increments* due to the switch back to NewReno.

Also, it can be seen with the CTCP *with muted dwnd increments* trace there are periods of noise between the two CTCP *with muted dwnd increments* flows as they attempt to achieve fair sharing. This contributes to slightly higher measured values of  $\sigma_f$  and  $\xi_f$ . The noise in question could be due to the sampling differences of RTT of each flow that may cause the *dwnd* dynamics to be less stable.

### 7.3 Eight Flows

Similar to the results with a single and two flows, the new algorithmic versions of CTCP shows improved performance for both increased throughput and better fairness between the flows. Again, the differences between the two are negligible.

Flows	CTCP Algorithm	Samples	Throughput	$\sigma_f$	$\xi_f$
1	<i>with muted dwnd increments</i>	5	119±19	–	–
1	<i>with DiffwndBasedFairness</i>	8	131±9	–	–
1	<i>with LossWindowBasedFairness</i>	2	103±33	–	–
2	<i>with muted dwnd increments</i>	19	157±12	0.08±0.02	0.33±0.01
2	<i>with DiffwndBasedFairness</i>	15	200±21	0.12±0.03	0.28±0.03
2	<i>with LossWindowBasedFairness</i>	18	182±17	0.15±0.02	0.30±0.02
8	<i>with muted dwnd increments</i>	10	377±17	0.17±0.02	0.51±0.01
8	<i>with DiffwndBasedFairness</i>	9	406±10	0.17±0.02	0.49±0.01
8	<i>with LossWindowBasedFairness</i>	8	318±87	0.13±0.03	0.47±0.04

Table 9: Performance of CTCP with  $\gamma$  auto-tuning from SLAC to Ireland.

## 8 $\gamma$ Auto-Tuning

CTCP requires a suitable value of  $\gamma$  to such that it can gather sufficient delay information from the network in order to operate effectively. This is true of all delayed based congestion control algorithms, and each individual flow (such as a single CTCP flow) needs to maintain approximately  $\gamma$  number of packets on the network to order for the flow to remain in equilibrium. All previous tests were conducted with a design parameter of  $\gamma = 30$  packets.

However, certain network links may not be able to maintain such large numbers of packets - especially as the number of flows scale. Furthermore, the optimal value of  $\gamma$  will differ depending upon the end-to-end network path and the volume of cross traffic that will affect the amount of buffering available.

The outcome of having too large a  $\gamma$  for a particular network path was seen in Figures 7 and 8.

In order to improve the interaction of CTCP under different network environments and to reduce the large bursts of packets due to a large static value of  $\gamma$ , an auto-tuning feature was implemented.  $\gamma$  auto-tuning facilitates the adaption of the  $\gamma$  value such that the delay part of the CTCP congestion control can function more appropriately under different network conditions.

Two algorithms were devised by Microsoft to enable this functionality. Both also implement the CTCP *with muted dwnd increments* variant described as above and were tested on the SLAC-Ireland path as shown in Table 9.

These tests were performed in the middle two weeks of May 2006.

### 8.1 Single Flow

It can be clearly seen that CTCP *with DiffwndBasedFairness* has the highest mean throughput, however, when the error is considered, all versions appear to perform similarly.

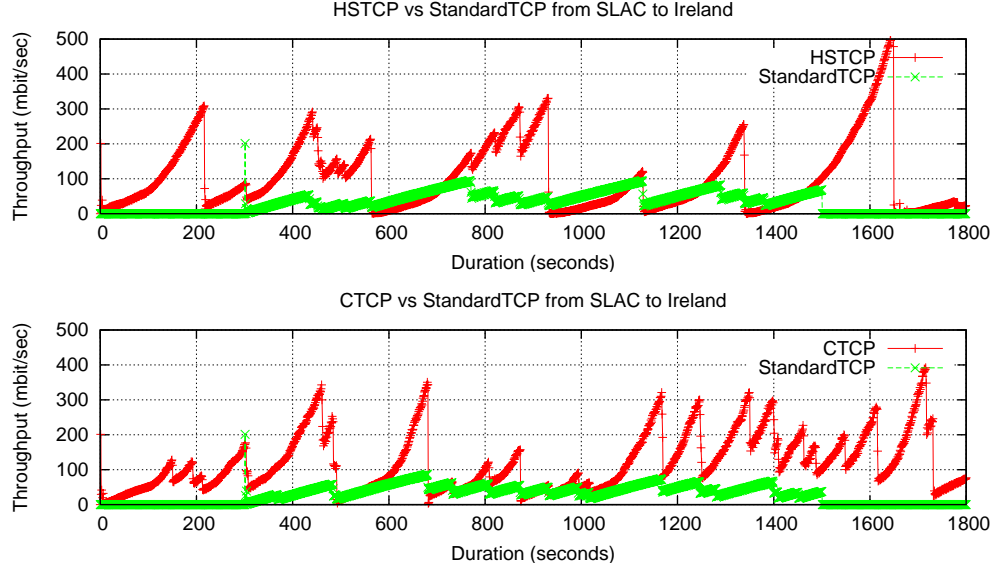


Figure 13: Friendliness of HSTCP and CTCP against StandardTCP.

## 8.2 Two Flows

When we move to two competing flows, it can be seen that both  $\gamma$ -tuned CTCP versions demonstrate marginally higher throughput compared to the non- $\gamma$  tuned version. However,  $\sigma_f$  is a little higher (although not statically significant) whilst  $\xi_f$  is consistent across all three versions.

## 8.3 Eight Flows

As we scale to more flows, it can be seen that the gamma-tuned CTCP algorithms demonstrate better  $\sigma_f$  and  $\xi_f$  compare to that of the CTCP *with muted dwnd increments* variant. Whilst throughput remains similar with or without the  $\gamma$  auto-tuning.

# 9 Friendliness

Understanding the interaction between the New-TCP algorithms and that of existing StandardTCP traffic (*friendliness*<sup>||</sup>) is useful to gauge the potential deployment issues of these new congestion control schemes.

Figure 13 illustrates the interaction between HSTCP and StandardTCP and also that of CTCP and StandardTCP from SLAC to Ireland. It can be seen that the effect of the CTCP upon the StandardTCP throughput is similar to that of HSTCP under this particular path.

<sup>||</sup>As per [4], the technical definition of *friendliness* is that of fairness but when there is a mix of new TCP flow(s) with StandardTCP flow(s). Similar to the definition of *fairness*, there are many points of view as to what *friendliness* constitutes. Therefore the definition of *friendliness* is intentionally vaguely defined.

	Ireland (Samples)	Florida
8 StandardTCP	310.68±21.44	340.80±4.23
Total	310.68±21.44 (6)	340.80±4.23 (9)
7 StandardTCP	182.91±16.41	290.65±2.48
1 CTCP	125.47±17.72	79.32±1.81
Total	312.31±33.36 (7)	371.83±0.65 (4)
7 StandardTCP	206.10±7.67	222.24±2.07
1 HSTCP	122.89±8.16	88.21±1.29
Total	332.29±17.33 (7)	310.97±0.88 (4)

Table 10: Aggregate Throughput (Mbit/sec) of Friendliness Tests.

Destination	New-TCP	Throughput per StandardTCP flow		Change
		Without New-TCP	With New-TCP	
Ireland	CTCP	38.83±2.68	26.13±2.34	-32%±7%
Ireland	HSTCP	38.83±2.68	29.44±1.16	-24%±6%
Florida	CTCP	42.60±5.29	41.52±0.35	-2%±1%
Florida	HSTCP	42.60±5.29	31.75±0.29	-25%±1%

Table 11: Impact of a single New-TCP flow upon StandardTCP flows.

However, it should be noted that in both cases, the problems of multiple consecutive drops (See Section 6.1) is apparent and therefore skews the results.

In an attempt to determine the friendliness, or rather, the *impact* of running these New-TCP algorithms against StandardTCP flows, we use the example of eight competing flows. All eight flows were initiated at the same time; a baseline measurement is taken whereby all eight flows are StandardTCP. We then compare the average throughput per flow against that of when we ‘swap’ one of these flows with that of a New-TCP algorithm (e.g. 7 StandardTCP flows and a single CTCP flow).

## 9.1 Burst Control

Table 10 and Table 11 shows the effect of switching one StandardTCP flow with that of CTCP (in this particular case, we are using CTCP *with muted dwnd increments*) and HSTCP against that of the baseline measurement of 8 StandardTCP flows.

On the well provisioned (in terms of buffering) Internet path to Florida, it can be seen that CTCP has almost no noticeable impact upon the StandardTCP per flow throughput, whilst HSTCP degrades the average throughput per StandardTCP flow by approximately -25%±1%. CTCP’s (lack of) impact also comes at the cost of a slightly reduced throughput performance; with the single CTCP achieving approximately 10% less throughput than that of HSTCP.

However, on the Internet path to Ireland, it can be seen that CTCP has slightly higher

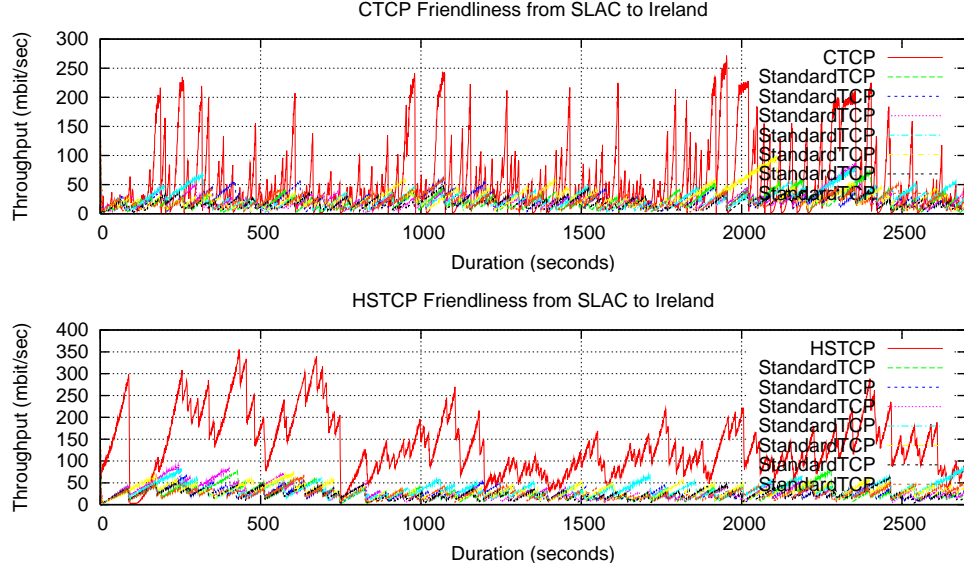


Figure 14: Friendliness of a single HSTCP and CTCP flow against 7 StandardTCP flows from SLAC to Ireland.

impact than HSTCP with upto a  $-32\% \pm 7\%$  reduction in the per StandardTCP flow throughput compared to that of  $-24\% \pm 6\%$  for HSTCP. Both individual CTCP and HSTCP flows achieve statistically indifferent throughputs under this path, but the HSTCP-StandardTCP aggregate is able to achieve slightly higher throughput on average (although it is not statistically significant).

For both network paths, it was observed that the throughput variation for CTCP is greater than that of HSTCP. The differences between CTCP and HSTCP for the Ireland path are shown in Figure 14. It can be seen that whilst CTCP is quick to reduce throughput and hence maintain low impact upon the competing StandardTCP flows, HSTCPs is slower to relinquish throughput with the effect of increased HSTCP throughput but reduced throughput per StandardTCP flow.

Under the SLAC-Florida path (See Figure 15), it can be seen that the CTCP flow is able to revert back to NewReno like behaviour which enables higher throughput per StandardTCP flow. This feature was not readily observed on the SLAC-Ireland path, and appears to result in more aggressive throughput increases which causes the StandardTCP flows to back off more and hence increases the measured (negative) impact which was observed.

## 9.2 $\gamma$ Auto-Tuning

In order to gauge the impact of the  $\gamma$  auto-tuning, tests were repeated to Ireland to determine the effect of the new algorithms upon StandardTCP. Table 12 and 13 show the results from the impact tests.

In previous tests, CTCP *with muted dwnd increments* demonstrated at  $32\% \pm 7\%$  impact upon the StandardTCP traffic, however, in these tests, CTCP *with muted dwnd increments*

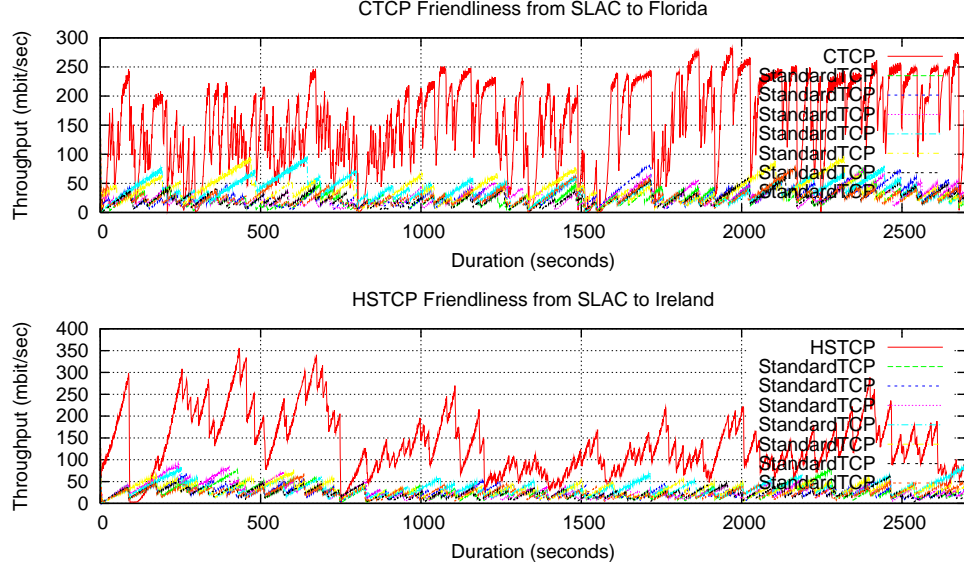


Figure 15: Friendliness of a single HSTCP and CTCP flow against 7 StandardTCP flows from SLAC to Florida.

	Ireland (Samples)
8 StandardTCP	186.903±11.013
Total	186.903±11.013 (7)
7 StandardTCP	152.688±16.422
1 CTCP with muted dwnd increments	82.758±8.574
Total	240.311±24.473 (9)
7 StandardTCP	167.908±15.638
1 CTCP with DiffwndBasedFairness	84.326±5.628
Total	252.361±20.471 (10)
7 StandardTCP	144.714±6.812
1 CTCP with LossWindowBasedFairness	75.610±4.637
Total	227.440±11.130 (10)

Table 12: Aggregate Throughput (Mbit/sec) of Friendliness Tests.

Destination	CTCP Algorithm	Throughput per StandardTCP flow		Change
		Without New-TCP	With New-TCP	
Ireland	<i>muted dwnd increments</i>	23.522±1.378	21.813±2.346	-7%±11%
Ireland	<i>DiffwndBasedFairness</i>	23.522±1.378	23.987±2.234	+2%±2%
Ireland	<i>LossWindowBasedFairness</i>	23.522±1.378	20.673±0.973	-12%±1%

Table 13: Impact of a single New-TCP flow upon StandardTCP flows.



only shows approximately  $7\% \pm 11\%$ . The impact of CTCP *with muted dwnd increments* in these tests demonstrate the varying nature of the network and the importance of having a large number of samples.

There appears to be no statistically significant difference between the throughputs achieved between the different versions, however CTCP *with LossWindowBasedFairness* does show a slightly lower throughput of  $76 \pm 5$  Mbit/sec compared to CTCP *with DiffwndBasedFairness*'s  $85 \pm 6$  Mbit/sec. What is perhaps more noticeable is the impact upon the per-flow StandardTCP throughput - with CTCP *with LossWindowBasedFairness* actually having a larger impact upon the StandardTCP traffic even though it does not appear to offer a significant increase in the throughput of the CTCP flow.

Generally, CTCP *with DiffwndBasedFairness* appears to be the best performing CTCP variant in these tests with negligible impact upon the StandardTCP traffic.

## 10 Discussion

The main concern is that all TCP tests exhibit performance problems related to the multiple consecutive drops. As such, the results are not in fact indicative of the performance of the TCP congestion control algorithm, but the TCP stack as a whole. Further investigation into the root cause of these drops is required to fully understand the consequence of deploying these new TCP algorithms; for example, is the cause of these drops due to aggressive re-transmission strategies that are magnified by the more aggressive growth rates of *cwnd*? or are they TCP stack implementation issues such as SACK processing?

The result of such problems is that whilst it appears as though CTCP performs well compared to the other TCP algorithms, in fact, it is not CTCP that is under investigation, but the ability for CTCP to quickly regain bandwidth after these episodes of *cwnd/ssthresh* reduction.

However, when analysed as a stack, rather than pure congestion control changes, CTCP appears to perform similarly to HSTCP in terms of throughput. The only exceptions are that when there is reverse traffic; however, a severe reduction of throughput is also apparent on the other stacks and further investigation is required.

In terms of fairness, as defined by  $\sigma_f$  and  $\xi_f$ , CTCP seems to perform somewhere between HSTCP and StandardTCP. On the Ireland link, where there appears to be insufficient buffering to enable CTCP to work efficiently, the fairness is actually worse than that of StandardTCP, and only comparable to that of HSTCP. On better provisioned (in terms of buffering) links such as the Caltech and Florida, the fairness is comparable, and sometimes even better than that of StandardTCP. Under network environments where there are small number of network buffers, it was found that the CTCP algorithm is unable to stabilise as effectively due to the relatively large value of  $\gamma$  used in the tests.

Therefore, two CTCP with burst control were tested that implement mechanisms to reduce the rate of *cwnd* increase when *diffWnd* is measured to be between  $\gamma_{low}$  and  $\gamma$ . Both versions gave better performance than that of the the original algoirthm and were demonstrated to work effectively on the buffer restricted SLAC-Ireland Internet path with both new versions demonstrating similar throughput and fairness characteristics. However,

upon inspection of the throughput traces, *CTCP with muted dwnd increments* appears to behave more closely to the expected CTCP dynamic, and suffers from fewer induced losses due to the successful reversion to the NewReno *cwnd* dynamic before congestion loss.

A concern with using delay sensitive congestion control protocols is their scaling property; how will it perform when there are hundreds or even thousands of flows? As demonstrated with *CTCP with muted dwnd increments* and the original CTCP algorithm under well buffered paths, the design of CTCP is such that it should fall back into StandardTCP's NewReno congestion control algorithm. However, CTCP often starts in a binominal increase of throughput such that it is able to utilise the seemingly available bandwidth. This relatively aggressive increase may have repercussions upon global synchronisation of Internet traffic and hence a general reduction in performance for all network users. Preliminary studies with the impact/friendliness show that under Internet paths with small buffers, CTCP does indeed demonstrate reduced fairness towards Internet traffic.

In an attempt to reduce this effect and to enable CTCP to function better on such paths, two versions of CTCP with  $\gamma$  auto-tuning were investigated to determine the effects of both intra and inter protocol fairness. Whilst there appeared to be no significant difference in performance when only CTCP flows were competing, *CTCP with DiffwndBasedFairness* demonstrated very little impact upon the throughput of competing StandardTCP traffic.

However, further investigation is warranted to fully understand this effect, with the possible tuning of the  $\gamma_{low}$  and  $\gamma$  parameters to better suit CTCP for more network environments.

## 11 Future Work

- **Stability analysis:** Given the delayed based adaptation of CTCP's congestion control to affect its' throughput, under different scenario's, CTCP may lead to better or worse variation of throughput through time. Whilst the presence of multiple consecutive drops in throughput is prevalent, a study of this property, commonly known as *smoothness*, of the CTCP algorithm may prove interesting.
- **Effect of different network conditions:** Related to  $\gamma$  settings, it may be useful to identify more sites around the world that will provide a broader depiction of the performance of CTCP. This should include tests at higher speeds (e.g. 10Gbit/sec - interesting due to the potential limitation of queue sizes across such scenario's), and also the effects of different latencies of competing flows.
- **Multiple Consecutive Drops:** Further investigation into this problem should be conducted to ensure that there are no issues with the TCP stack implementation and the effect is actually real and expected.
- **Convergence Times:** due primarily to the (lack of) resolution in the iperf logs, it was not possible to determine accurately the convergence times of the flows.

In previous studies [4], we defined the convergence time at the time required for the throughputs between two flows to converge to some arbitrary ratio. This was particularly useful to determine the transient properties of interacting traffic.

- Effect of Reverse Traffic: Tests from SLAC to Florida and Caltech show reduced performance of a single TCP flow in the presence of reverse traffic compared to that without. Some further investigation into the cause of this is required.
- Impact on existing traffic: whilst the study of induced traffic to determine the effect of CTCP is warranted, it may be possible to gather queue dropping statistics from certain backbone providers such as ESnet and Internet2. The advantage of this is that it will be possible to analyse the real effect upon existing Internet users of deploying new TCP algorithms.
- Mice versus Elephant Traffic: whilst this investigation is limited to long lived flows, it would be expected that much of the traffic will be relatively short lived. Some investigation into this dynamic should be performed to determine whether the aggressive increase of CTCP will affect network stability.

## 12 Summary

This work serves as an initial field study of the performance of CTCP on the Internet. Due to the adaptation of only the congestion control algorithms for CTCP's implementation, there is expected to be no compatibility problems associated with its deployment.

The concerns of CTCP deployment are such that it does not affect the scalability and stability of the Internet.

This initial field study demonstrated that there are TCP stack issues (rather than congestion control issues) that affect the performance of the TCP tests - these were visible with the occurrence of multiple consecutive drops and the unexplained reduction of *ssthresh* to very small values with the Windows Vista TCP stack.

In general, CTCP appears to perform somewhere between HSTCP and StandardTCP; its performance appears to be related to the queue provisioning on the network path. On well provisioned networks such as to Caltech and Florida, CTCP demonstrates good fairness in terms of  $\sigma_f$  and  $\xi_f$ . However, on the Ireland link - which is known to have relatively small network buffering, tests show that the performance is comparable to HSTCP.

In order to improve the performance of CTCP on buffer restricted Internet paths, two versions of CTCP with burst control were also tested which has a less aggressive response under more restrictive network environments. Initial trials of these algorithms proved successful in improving both the throughput and fairness metrics compared to that of the original CTCP algorithm, and are recommended to be included as part of the standard CTCP deployment.

An preliminary study of the impact of running a single CTCP flow in the presence of many StandardTCP flows was also conducted. Under the medium distance Florida path, CTCP performs better than HSTCP in terms of impact at the cost of having slightly reduced throughput (by about 10%). Under the Internet path to Ireland, there was a small but noticeable increase in the impact when using CTCP compared to HSTCP (approximately 10% greater reduction in per flow StandardTCP throughput).

In order to reduce the impact of the CTCP flows against StandardTCP, two CTCP versions incorporating  $\gamma$  auto-tuning were also investigated that demonstrated insignificant

performance differences over CTCP *with muted dwnd increments* in intra-protocol tests. However, CTCP *with DiffwndBasedFairness* demonstrated very good results in almost having zero impact upon competing StandardTCP flows on the limited test environment of the SLAC to Ireland path.

In order to better understand the effects of CTCP deployment, we suggest further tests to be developed to investigate the scaling properties of CTCP, especially under relatively small queue sizes, and more thorough tests to understand and judge the impact of CTCP traffic against other Internet users. In particular, more generic parameters of  $\gamma_{low}$  and  $\gamma$  should be investigated under various network environments to better tune CTCP.

## Acknowledgments

We wish to thank Microsoft for their software and engineering support during the investigation.

Neterion were also a great help during our time of need to get our NICs running on our Windows Longhorn testbed.

We would also thank the people at Caltech, University of Florida and the Hamilton Institute for their contribution of hardware to make these tests possible.

## References

- [1] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-speed Networks," in PFLDNet, Japan, February 2006.
- [2] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A Compound TCP Approach for High-Speed and Long Distance Networks," in INFOCOMM, Barcelona, Spain, April 2006.
- [3] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. Iperf version 1.7.0. <http://dast.nlanr.net/Projects/Iperf/>, March 2003.
- [4] Y. Li, D. Leith, and R. Shorten. Experimental Evaluation of TCP Protocols for High-Speed Networks. Submitted to IEEE/ACM Transactions on Networking, June 2005.
- [5] R. King, R. Riedi, and R. G. Baraniuk. Tcp-africa: An adaptive and fair rapid increase rule for scalable tcp. In IEEE Infocom, march 2005.
- [6] S. Floyd, HighSpeed TCP for Large Congestion Windows. RFC 3649, Experimental, December 2003.